

CLAIMS

What is claimed is:

1. A method comprising:
 - dispatching an instruction for execution;
 - speculatively executing said instruction;
 - determining whether said instruction executed correctly;
 - routing said instruction to a replay mechanism if said instruction did not execute correctly;
 - determining whether incorrect execution of said instruction is due to a long latency operation;
 - routing said instruction for immediate re-execution if said incorrect execution is not due to said long latency operation;
 - delaying routing of said instruction for re-execution if said incorrect execution is due to said long latency operation;
 - re-executing said instruction if said instruction did not execute correctly; and
 - retiring said instruction if said instruction executed correctly.
2. The method of claim 1 wherein said routing said instruction comprises advancing said instruction forward for replay into an earlier replay time slot if a slot comes available.
3. The method of claim 2 wherein said delaying routing of said instruction comprises moving said instruction backwards in time as resource conflicts are detected.
4. The method of claim 3 further comprising:
 - routing said instruction to a delay queue to wait a period of time before re-executing said instruction if said incorrect execution is due to said long latency operation;
 - determining what type of error caused said instruction to execute incorrectly; and
 - predicting what length time period to delay said instruction prior to routing said instruction for re-execution if said incorrect execution is due to said long latency

8 operation.

1 5. The method of claim 4 further comprising tracking number of times said
2 instruction is executed and re-executed.

1 6. The method of claim 5 further comprising increasing said time period to delay
2 said instruction for re-execution as number of time said instruction has been executed and
3 re-executed increases.

1 7. The method of claim 6 wherein said retiring further comprises applying a result of
2 said instruction to an architectural state if said instruction executed correctly.

1 8. The method of claim 7 further comprising discarding execution result of said
2 instruction if said instruction executed incorrectly.

1 9. A method comprising:
2 dispatching an instruction;
3 speculatively executing said instruction;
4 checking whether said instruction executing correctly;
5 replaying said instruction if said instruction did not execute correctly, wherein
6 said replaying comprises dynamically determining a time period to delay said
7 instruction prior to re-execution;
8 re-executing said instruction after said time period elapsed if said instruction did
9 not execute correctly; and
10 retiring said instruction if said instruction executed correctly.

1 10. The method of claim 9 wherein said checking further comprises determining
2 whether a long latency type of error caused said instruction to execute incorrectly if said
3 instruction did not execute incorrectly.

1 11. The method of claim 10 wherein said dynamically determining said time period

2 comprises shifting said instruction forward for earlier replay if a time slot become
3 available and shifting said instruction backwards for later replay if a resource conflict is
4 detected.

1 12. The method of claim 11 further comprising:
2 routing said instruction to a delay queue to wait for said time period to elapse
3 prior to re-executing said instruction if said long latency type of error caused said
4 instruction to execute incorrectly; and
5 routing said instruction from said delay queue for re-execution after said time
6 period has elapsed.

1 13. The method of claim 12 wherein said routing of said instruction from said delay
2 queue comprises rescheduling said instruction for re-execution.

1 14. The method of claim 12 wherein said routing of said instruction from said delay
2 queue comprises sending said instruction for re-execution without rescheduling said
3 instruction.

1 15. The method of claim 12 further comprising tracking number of times said
2 instruction is executed and re-executed, wherein said time period to delay said instruction
3 for re-execution is increased as number of times said instruction has been executed and
4 re-executed increases.

1 16. The method of claim 15 wherein said retiring further comprises applying a result
2 of said instruction to an architectural state if said instruction executed correctly.

1 17. A processor comprising:
2 a scheduler to dispatch instructions;
3 a multiplexor coupled to said scheduler, said multiplexor to receive said
4 instructions from said scheduler;
5 an execution unit coupled to said multiplexor, said execution unit to execute said

FOOTNOTES

6 instructions;
7 a checker coupled to said execution unit, said checker to determine whether each
8 instruction has executed correctly; and
9 a replay mechanism coupled to said checker, said replay mechanism to receive
10 from said checker each instruction that has not executed correctly, said replay
11 mechanism further comprising logic to determine whether a long latency operation
12 caused an incorrectly executed instruction, said logic also to dynamically determine a
13 time period to delay said incorrectly executed instruction if said long latency
14 operation caused said incorrectly executed instruction.

1 18. The processor of claim 17 wherein said replay mechanism further comprises a
2 delay queue to store said incorrectly executed instruction for said time period prior to
3 releasing said incorrectly executed instruction to said execution unit for re-execution.

1 19. The processor of claim 18 wherein said replay mechanism further comprises a
2 counter to count a number of times said incorrectly executed instruction is executed and
3 re-executed.

1 20. The processor of claim 19 wherein said logic increases said time period to delay
2 said incorrectly executed instruction as said number of times increases.

1 21. The processor of claim 20 further comprising:
2 a retirement unit coupled to said checker to receive any instructions that have
3 executed correctly, said retirement unit to retire said instructions that have executed
4 correctly;
5 a first level internal cache coupled to said execution unit; and
6 a second level internal cache coupled to said execution, wherein access time to
7 said second level internal cache is greater than to said first level cache.

1 22. The processor of claim 21 wherein said processor is coupled to an external main
2 memory and to a disk memory.

1 23. The processor of claim 22 wherein said incorrectly executed instruction is a
 2 memory load operation, said memory load instruction operation to cause a memory fetch,
 3 said memory fetch to search through a memory hierarchy for requested data, wherein said
 4 memory hierarchy comprises of said first level cache, said second level cache, said
 5 external main memory and said disk memory, and wherein said first level cache has
 6 fastest access time and said disk memory has longest access time.

1 24. The processor of claim 23 wherein each incorrect execution of said memory load
 2 instruction causes said memory fetch to access a slower level of memory, and said logic
 3 increases said time delay to approximate an access latency to said slower level of
 4 memory.

1 25. The processor of claim 17 wherein said replay mechanism comprises an
 2 advance/delay queue to shift said incorrectly executed instruction forward if a time slot
 3 opens up and to shift said incorrectly executed instruction backward if a resource conflict
 4 is detected.

1 26. A system comprising:
 2 a memory coupled to a bus;
 3 a processor coupled to said bus, said processor comprising:
 4 a scheduler to dispatch instructions;
 5 an execution unit coupled to said scheduler, said execution unit to execute
 6 said instructions;
 7 a first level cache coupled to said execution unit, said first level cache to
 8 store data for said instructions;
 9 a checker coupled to said execution unit, said checker to determine
 10 whether each instruction has executed correctly; and
 11 a replay mechanism coupled to said checker, said replay mechanism to
 12 receive from said checker each incorrectly executed instruction, said replay
 13 mechanism further comprising logic to determine whether a long latency
 14 operation caused said incorrectly executed instruction, said logic to also

15 dynamically determine a time period to delay said incorrectly executed
16 instruction if said long latency operation caused said incorrectly executed
17 instruction.

1 27. The processor of claim 26 wherein said replay mechanism comprises an
2 advance/delay queue to shift said incorrectly executed instruction forward for earlier
3 replay if a time slot opens up and to shift said incorrectly executed instruction backward
4 for later replay if a resource conflict is detected.

1 28. The system of claim 26 wherein said replay mechanism further comprises a delay
2 queue to store said incorrectly executed instruction for said time period, said delay queue
3 to release said incorrectly executed instruction to said execution unit for re-execution
4 after said time period has elapsed.

1 29. The system of claim 28 further comprising a retirement unit coupled to said
2 checker, said retirement unit to retire each instruction receive from said checker that
3 executed correctly.

1 30. The system of claim 29 wherein said replay mechanism further comprises a
2 counter to track a number of times said incorrectly executed instruction is executed and
3 re-executed, said logic to increase said time period to delay said incorrectly executed
4 instruction as said number increases.